Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Teja Roštan

Rezanje nevronskih mrež z matrično faktorizacijo

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Curk

Ljubljana 2015

Rezultati diplomskega dela so intelektualna lastnina avtorja. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite metode rezanja (angl. *pruning*) umetnih nevronskih mrež. Preučite, kako uporabiti metode matrične faktorizacije, predvsem metode hkratne faktorizacije več matrik za rezanje nevronskih mrež in predlagajte novo metodo rezanja. Na več naborih podatkov ovrednotite in poročajte o napovedni uspešnosti in računski zahtevnosti nerezanih nevronskih mrež in mrež pridobljenih z različnimi metodami rezanja. Primerjajte in poročajte o uspešnosti obstoječih metod rezanja nevronskih mrež in predlagane metode.

Implement and test standard methods for pruning artificial neural networks. Develop a novel method for pruning that is based on simultaneous matrix trifactorization. Use a number of data sets to empirically evaluate the predictive performance and computational demands of unpruned and pruned artificial neural networks. Compare the performance of standard pruning methods and the proposed pruning method.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Teja Roštan sem avtorica diplomskega dela z naslovom:

Rezanje nevronskih mrež z matrično faktorizacijo

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Tomaža Curka,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 7. septembra 2015 Podpis avtorja:

Zahvaljujem se mentorju, doc. dr. Tomažu Curku, za vso znanje, ki sem ga pridobila pod njegovim mentorstvom. Zahvaljujem se tudi svojim najbližjim, ker so verjeli vame. Posebna zahvala gre mojemu fantu, ki mi je bil vedno v oporo. Nenazadnje se zahvaljujem tudi svojim sošolcem, ki so poskrbeli za prijetno vzdušje tekom študija.

Svoji dragi družini in Žigu.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AUC	area under curve	površina pod krivuljo ROC
ROC	receiver operating characteristic	krivulja ROC
ANN	artificial neural network	umetna nevronska mreža
MF	matrix factorization	matrična faktorizacija
MFBP	matrix factorization-based brain pruning	rezanje možganov z matrično faktorizacijo
MFBP1	matrix factorization-based brain pruning, first criterium	rezanje možganov z matrično faktorizacijo, prvi kriterij
MFBP2	matrix factorization-based brain pruning, second criterium	rezanje možgranov z matrično faktorizacijo, drugi kriterij
NMF	non-negative matrix factorization	nenegativna matrična faktorizacija
DNN	deep neural network	globoka nevronska mreža
CNN	convolutional neural network	konvolucijske nevronske mreže
RNN	recurrent neural network	rekurenčne nevronske mreže
BPNN	backpropagational neural network	nevronska mreža z vzvratnim pravilom učenja
SVM	support vector machine	metoda podpornih vektorjev
RMSE	relative mean square error	relativna povprečna kvadratna napaka
OBD	optimal brain damage	optimalna poškodba možganov
OBS	optimal brain surgeon	optimalen možganski kirurg
MBP	magnitude based pruning	magnitudno osnovano rezanje
SKL	skeletonization	skeletonizacija
SNNS	Stuttgart neural network simulator	Stuttgartova simulacija nevronske mreže

Povzetek

Matrično faktorizacijo, ki se povezuje s postopkom zlivanja podatkov, uporabljamo za odkrivanje vzorcev oziroma skupin v podatkih. Faktorizirani model preslika podatke v nižje-dimenzionalen prostor, jih tako skrči in odpravi del šuma. Tovrstni modeli so zato navadno bolj robustni in imajo višjo napovedno točnost. Pri nevronskih mrežah bi tako znali reševati problem prevelike prilagojenosti podatkom (angl. *overfitting*) in pridobili pri generalizaciji. V nalogi smo preučili, ali s hkratno faktorizacijo parametrov nevronske mreže, ki jih je možno predstaviti z več matrikami, odstranimo (porežemo) nepomembne povezave in tako izboljšamo napovedno točnost mreže. Predlagani postopek rezanja smo preizkusili na navadnih in globokih nevronskih mrežah. Po uspešnosti je primerljiv z ostalimi najuspešnejšimi standardnimi pristopi rezanja nevronskih mrež.

Ključne besede: nevronske mreže, matrična faktorizacija, rezanje.

Abstract

Matrix factorization and the procedure of data fusion are used to detect patterns in data. The factorized model maps the data to a low-dimensional space, therefore shrinking it and partially eliminating noise. Factorized models are thus more robust and have a higher predictive accuracy. With this procedure we could solve the problem of overfitting in neural networks and improve their ability to generalize. Here, we report on how to simultaneously factorize the parameters of a neural network, which can be represented with multiple matrices, to prune not important connections and therefore improve predictive accuracy. We report on empirical results of pruning normal and deep neural networks. The proposed method performs similarly to the best standard approaches to pruning neural networks.

Keywords: neural networks, matrix factorization, pruning.

Poglavje 1

Uvod

Živimo v dobi zajemanja in procesiranja velikih količin podatkov. Zahteve, da iz teh podatkov pridobimo informacije in novo znanje, so vedno večje. Posledica je razvoj orodij analize podatkov in strojnega učenja. Velike količine podatkov so torej ustvarile novo zanimanje za predstavitev in zmanjšanje dimenzionalnosti podatkov [19]. Pri tem na vedno večji veljavi pridobivata pristopa nevronskih mrež in matrične faktorizacije. Njuno uporabno vrednost najdemo v znanosti, gospodarstvu, zabavni industriji in medicini. Tako so primeri uporabe nevronskih mrež napoved bankrota podjetja, medicinska diagnoza, kontrola kvalitete proizvodov, prepoznavanje ročno napisanega besedila, procesiranje slik in prepoznavanje govora. Primeri uporabe matrične faktorizacije pa so prepoznavanje vzorcev, multimedija, tekstovno rudarjenje in analize biomedicinskih podatkov [25].

1.1 Napovedovanje z nevronskimi mrežami

Navadne umetne nevronske mreže (ANN) imajo ponavadi enega do dva skrita nivoja, medtem ko imajo globoke nevronske mreže (DNN) več skritih nivojev. Globoko učenje deluje zaradi arhitekture mreže in njene optimizacije [14, 20, 37]. Nevronsko mrežo predstavimo z usmerjenim grafom. Vsaka skrita enota je tako povezana z veliko ali vsemi prejšnjimi skritimi enotami.

Torej je vsak skriti nivo globje v mreži nelinearna kombinacija nivojev pred njim, in sicer zaradi vseh kombinacij in rekombinacij izhodov iz vseh prejšnjih enot, ki so v kombinaciji z njihovimi aktivacijskimi funkcijami. Z učenjem na podatkih vsak skriti nivo postane optimalno uteženа, nelinearna kombinacija nivojev pred njim. Vsak naslednji skriti nivo pa postane projekcija nižje dimenzije. Informacija prejšnjega nivoja je torej povzeta z nelinearno, optimalno uteženo, nižje-dimenzionalno projekcijo v vsakem naslednjem nivoju v globoki mreži [15, 32]. Z globokimi mrežami tako zgradimo bolj kompleksne hierarhije in hkrati pridobimo na robustnosti [15].

Globoke mreže so zaradi svoje velike arhitekture prispevale k različnim načinom razvoja mrež, predvsem k metodam nadzorovanega učenja. Med globokimi mrežami z nadzorovanim učenjem so se najbolj uveljavile konvolucijske globoke mreže (angl. *convolutional neural network* (CNN)) [21]. Njene lastnosti so pomembne predvsem pri prepoznavanju vzorcev, kot so lokalno dojemljivo polje (angl. *local receptive field*), skupne uteži in pristranskost (angl. *bias*), združevalni nivoji (angl. *pooling layers*) in max-združevanje (angl. *max-pooling*). Za naravni jezik pa so se razvile tako imenovane rekurenčne nevronske mreže (angl. *recurrent neural networks* (RNN)) [21]. Ker globoke mreže gradijo bolj kompleksne modele, jim ekspresivnost, fleksibilnost in robustnost učnih algoritmov omogočajo učenje kompleksnih predstavitev objektov, in sicer brez potrebe po ročnemu določanju in gradnji atributov. Globoke mreže se tako vedno bolj uveljavljajo in implementirajo v pametne telefone, računalnike, avtomobile, gospodinjske aparate in ostalo elektroniko.

1.2 Rezanje nevronskih mrež

Ko je učni sistem naučen na osnovi učnih primerov, je pomembno, da dobro napoveduje vzorce, ki so zunaj učne množice. Optimalna arhitektura mreže je dovolj velika, da zagotavlja, da se mreža nauči problema, in dovolj majhna, da mreža dobro generalizira. Mrežo optimalne velikosti bi lahko našli na

način, da bi naučili več mrež različnih velikosti in izbrali najmanjšo, ki se bo še naučila iz danih učnih podatkov. Ta način je seveda potraten, saj od nas zahteva, da moramo naučiti veliko mrež, preden najdemo primerno.

Problem izbire prave velikosti nevronske mreže rešujemo z dvema inkrementalnima pristopoma. Prvi začne z majhno mrežo in postopoma dodaja nove skrite enote ali nivoje, dokler uspešnost učenja ni zadovoljiva. Znan primer takšnih algoritmov je kaskadna korelacija. Drugi pristop je rezanje (angl. *pruning*), ki začne z veliko mrežo, ki jo sestavljajo utežene povezane enote, ki se ji potem odstrani nepotrebne uteži in/ali enote [17]. Problem prvega pristopa je v tem, da majhna mreža zaradi premajhnega števila nevronov morda nikoli ne bo rešila problema. Poleg tega je prednost globoke mreže še v tem, da se hitreje uči. Raziskave kažejo, da je v mnogih primerih celoten čas, potreben za učenje velike mreže in rezanje v manjšo mrežo, primerljiv z učenjem preproste majhne mreže [1, 36]. Zato lahko učenje in generalizacijo večjih mrež izboljšamo z rezanjem. Ravno ta način smo uporabili v pričujoči diplomski nalogi, pri čemer smo za rezanje uporabili metodo, ki temelji na matrični faktorizaciji nizkega ranga.

1.2.1 Boljše napovedi?

Umetne nevronske mreže pogosto dosežejo visoko klasifikacijsko točnost in so znane po svojih klasifikacijskih sposobnostih. Uspešnost nevronskih mrež je odvisna od tega, kako dobro napoveduje iz podatkov, ki so zunaj učne množice oziroma kako dobro generalizira. Generalizacijske zmožnosti umetne nevronske mreže so odvisne od velikosti učnih podatkov, števila učnih iteracij (angl. *epochs*) in arhitekture mreže. V literaturi [1] je bilo predstavljenih veliko različnih rezalnih algoritmov za optimizacijo arhitekture nevronskih mrež. Glavna ideja rezalnih algoritmov je, da odstranijo nepomembne ali škodljive povezave iz gosto povezanih nevronskih mrež. Ker rezane nevronske mreže vsebujejo manj parametrov, so manj nagnjene k preveliki prilagojenosti (angl. *overfitting*). Pristopi rezanja tako predvidevajo, da je dovolj velika nevronska mreža že naučena in se jo lahko poenostavi s sprejemljivo izgubo

klasifikacijske točnosti na učni množici, kar lahko vpliva na boljše napovedi.

1.2.2 Manjša računska zahtevnost?

Pri nevronskih mrežah meja kompleksnosti (angl. *VC dimension*) ni popolnoma jasna, zato je zelo pomembno, da pretehtamo, kako dobra je lahko rešitev. Največji problem nevronskih mrež je tako imenovana "črna škatla" (angl. *black box problem*). Le-ta se nanaša na to, da težko razložimo, kako se je učenje glede na vhodne podatke izvedlo. Mreža ne zna interpretirati povezav med vhodnimi in izhodnimi podatki in se ne zna soočiti z nedoločenostjo [13].

Majhne mreže imajo tako poleg boljše pričakovane generalizacije tudi to prednost, da je njihovo delovanje lažje razumeti, saj je manj možnosti, da bi mreža razširila funkcije preko več nevronov. To je pomembno v kritičnih aplikacijah, kjer mora uporabnik vedeti, kako sistem deluje. Tudi s tem, ko mrežo večje dimenzije porežemo, ji dimenzijo zmanjšamo in povečamo možnost interpretacije. Manjše mreže imajo večji pomen tudi pri mobilnih aplikacijah, saj zahtevajo manj računanja in manj pomnilnika. S tem se večina raziskovalcev ne ukvarja, saj je zanje običajno pomembnejše dobro napovedovanje, poraba sistemskih virov pa je drugotnega pomena. Z namenom zmanjšanja računske zahtevnosti smo uporabili matrično faktorizacijo nižjega ranga, kjer mora biti rang strogo manjši od dimenzij uteži v mreži. Na ta način zmanjšamo dimenzijo mreže.

Matrična faktorizacija je tehnika aproksimiranja matrike s pomočjo razcepljanja na produkt več manjših matrik (ponavadi dveh ali treh). Njen namen je, da najdemo približek izvorne matrike, s pomočjo odkritih latentnih faktorjev zmanjšamo dejansko dimenzionalnost podatkov in ohranimo le bistvene povezave.

1.3 Cilji

V diplomski nalogi želimo odgovoriti predvsem na naslednja vprašanja:

- Katerim tipom nevronskih mrež (navadne ali globoke) rezanje bolj pripomore?
- Ali rezanje z matrično faktorizacijo izboljša napovedno točnost nevronske mreže?
- Kako uspešno je rezanje z matrično faktorizacijo v primerjavi s standardnimi pristopi rezanja?
- Kako se stopnja rezanja odraža v računski in prostorski zahtevnosti ter uspešnosti mreže?

Poglavje 2

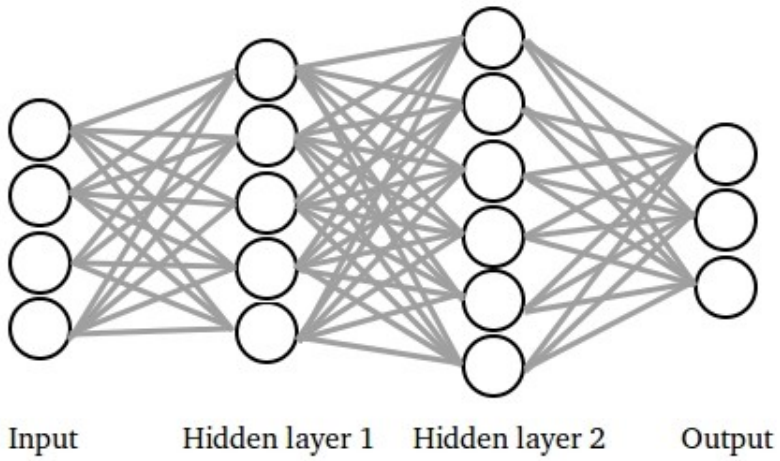
Metode in podatki

V tem poglavju sledi opis umetnih nevronskih mrež in obstoječih pristopov rezanja nevronskih mrež. Predlagan je nov postopek rezanja, ki temelji na hkratni faktorizaciji več matrik. Opisane so metode preverjanja napovedne uspešnosti in uporabljene podatkovne zbirke, ki smo jih uporabili za vrednotenje navedenih pristopov.

2.1 Umetne nevronske mreže

Umetne nevronske mreže so močno računsko orodje za delo na velikih količinah podatkov, ki ga ni težko implementirati, uporabljati in razumeti. Prve umetne nevronske mreže so bile zgrajene leta 1943. Definirala sta jih nevropsiholog Warren McCulloch in logik Walter Pits [34]. Koncept nevrona sta opisala kot biološko nevronsko celico, živečo v mreži enot, ki prejemajo vhodne podatke, jih procesirajo in generirajo izhod. Nevronske mreže so organizirane v nivoje, kot je prikazano na sliki 2.1. Nivoje določajo množice notranje povezanih enot, ki vsebujejo aktivacijsko funkcijo (npr. sigmoid). Vzorec vhodnih podatkov damo na vhodni nivo (angl. *input layer*), ki komunicira z enim ali več skritimi nivoji (angl. *hidden layer*). Računanje na skritih nivojih se izvaja na osnovi uteženih povezav. Zadnji skriti nivo je povezan z izhodnim nivojem (angl. *output layer*), kjer je izhod odgovor, ki

ga želimo izvedeti. Enot v vhodnem nivoju je torej toliko, kot je atributov v podatkovni zbirki, medtem ko je enot v izhodnem nivoju toliko, kot je vseh možnih ciljnih razredov. Število skritih nivojev in uporabljenih enot na vsakem skritem nivoju pa uporabnik določi sam glede na kompleksnost problema in podatkovne zbirke.



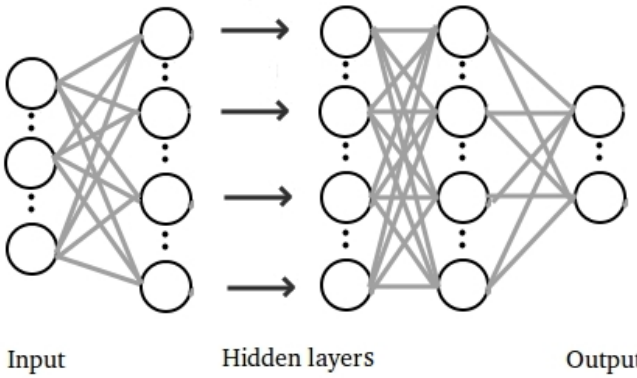
Slika 2.1: Nevronska mreže z vhodnim, izhodnim in dvema skritima nivojema, prvi s pet in drugi s šest enotami (vozlišči). Graf je gosto povezan.

Mreže tako predstavljajo sistem notranje povezanih nevronov (enot), ki si med seboj pošiljajo informacije. Uteži povezav se spreminjajo glede na izkušnjo (učne primere), ki te mreže doletijo. Njihov namen ni, da čim bolj natančno opišejo, kako delujejo biološki možgani, ampak so bile ustvarjene kot računski model za reševanje nekaterih vrst problemov. Nevronske mreže se kot metoda podatkom prilagodijo brez uporabnikove pomoči in posebnih specifikacij. So tudi univerzalni aproksimatorji, saj lahko aproksimirajo katerokoli funkcijo s poljubno majhno napako. Tako se pogosto uporabljajo za oceno ali približek funkcij, ki so lahko odvisne od velikega števila vhodov in so neznane. Pomembna značilnost predvsem globokih nevronskih mrež je, da znajo ujeti tudi nelinearne in kompleksne temeljne lastnosti kateregakoli procesa z visoko stopnjo točnosti, kar jih dela fleksibilne pri modeliranju kompleksnih razmerij.

Pomembna lastnost nevronske mreže je pravilo oziroma funkcija, ki določa, kako spremeniti uteži povezav glede na vhodne učne podatke. Čeprav obstaja veliko različnih vrst pravil za učenje, ki jih nevronske mreže uporabljajo, mreža ponavadi uporablja pravilo delta (angl. *delta rule*) [29]. To pravilo je uporabljeno pri najbolj popularni nevronski mreži, imenovani nevronska mreža z vzvratnim pravilom učenja (angl. *back-propagation neural network* (BPNN)). Te mreže so popularne zaradi strukturne fleksibilnosti in dobre reprezentacijske zmožnosti. Njihova značilnost je, da vzvratno popravljajo napako. Z delta pravilom je učenje nadzorovan proces, ki se pojavi z vsakim ciklom (angl. *epoch*), pri čemer je mreža vsakič predstavljena z novim vhodnim učnim primerom. Informacija najprej teče skozi aktivacijske funkcije od vhoda do izhoda, potem pa z vzvratnim odpravljanjem napak prilagaja uteži povezav. Nevronska mreža na začetku naključno ugiba, kaj bi lahko bil rezultat. Z vzvratnim popravljanjem napak pa ustrezno prilagodi uteži povezav in tako se napovedovanje pričakovanega rezultata izboljša.

Ko je nevronska mreža naučena do zadovoljive mere, jo lahko uporabimo kot analitično orodje nad ostalimi podatki. V tem primeru uporabniku mreže ni več potrebno učiti, ampak jo lahko uporabi za napovedovanje z uporabo načina razširjanja naprej. Novi vhodni podatki so predstavljeni na vhodnem nivoju, kjer se preslikajo skozi skrite nivoje glede na utežene povezave in aktivacijske funkcije, podobno kot v času učenja, dokler ne dosežejo izhodnega nivoja. Tako smo dobili napovedni model, ki ga lahko uporabimo za napovedovanje, analize in interpretacije.

V nedavnih obširnih raziskovalnih aktivnostih o nevronski klasifikaciji [39] so dokazali, da so nevronske mreže lahko obetavna alternativa številnim običajnim klasifikacijskim metodam. Najbolj pogosti problemi, s katerimi se sedaj soočajo nevronske mreže, so tako imenovani problemi tipa “lahki za človeka, težki za računalnik:” prepoznavanje vzorcev, obrazov, ročno napisanih pisav, govora, predmetov, slik ter priporočilni sistemi.



Slika 2.2: Globoka nevronska mreža po zgledu [2]. Gre za gosto povezan graf z mnogimi vozlišči (enotami) in skritimi nivoji.

2.1.1 Globoke nevronske mreže

Globoke nevronske mreže so zaradi svoje velike arhitekture (glej sliko 2.2) pripomogle k razvoju različnih modelov nevronskih mrež. Med njimi sta se najbolj razvili konvolucijska in rekurenčna nevronska mreža [9, 21].

Konvolucijska nevronska mreža se uporablja predvsem za prepoznavanje vzorcev na slikah. Pri tem se izkoriščajo nekatere pomembne lastnosti vhodnih slik. Ena od teh lastnosti je lokalno dojemljivo polje (angl. *local receptive field*), kjer se vhodne piksle lokalno poveže v enote v skritih nivojih. Ta lastnost je pomembna predvsem za prepoznavanje kotov in oblik na slikah. Druga pomembna lastnost, ki je značilna za konvolucijske globoke mreže, so skupne uteži in pristranskost (angl. *bias*), kar pomeni, da vsi nevroni v prvem skritem nivoju zaznajo povsem enake značilnosti, le da na različnih lokacijah vhodne matrike. Mreža lahko, na primer, razbere nek oster kot v določenem lokalnem zaznavnem polju. Ta sposobnost mreže je uporabna tudi na drugih lokacijah slike. Zato je priporočljivo, da se povsod na sliki vključi enake detektorje značilnic, ki so odporne na translacijsko invarianco slike. Preslikave iz vhodnega v prvi skriti nivo se zato pogosto poimenuje preslikave atributov. Tretja pomembna lastnost so združevalni nivoji (angl. *pooling layers*). Ti nivoji so takoj za konvolucijskimi nivoji in povzemajo informacije iz teh

nivojev. Združevalni nivoji vzamejo vsako izhodno preslikavo atributov iz konvolucijskega nivoja in pripravijo zgoščeno preslikavo. Naloga združevanja je, da združi semantično podobne atribute v enega. Ker konvolucijski nivo ponavadi vsebuje več preslikav atributov, uporabimo max-združevanje (angl. *max-pooling*) za vsako preslikavo atributov posebej.

Konvolucijske nevronske mreže so tako bolj prilagojene za klasifikacijo slik. Globoka nevronska mreža je močen model, ki dosega odlične uspehe pri težkih problemih prepoznavanja vzorcev v vidu in govoru. Za naravni jezik so se razvile rekurenčne nevronske mreže, ki v vhod sekvenčno prejemajo podatke, kot je govor. Tudi te so pokazale dobre rezultate, zato lahko v prihodnosti pričakujemo nove nevronske mreže, ki bodo kombinacija obeh tipov mrež.

2.2 Rezanje nevronskih mrež

Želimo naučiti napovedni model, ki bo dobro opisoval dane učne primere, hkrati pa bo dobro napovedoval (generaliziral) tudi nove primere izven učne množice. Učenje in generalizacija sta verjetno najbolj pomembni temi v raziskavah nevronskih mrež. Učenje je zmožnost, da aproksimiramo podano vedenje s prilagajanjem modela k učnim podatkom. Generalizacija je zmožnost, da napovedujemo onstran učnih podatkov. Če je učna množica nepopolna ali premajhna, ima lahko lažne in zavajajoče vzorce, ki dejansko predstavljajo šum [33]. Na neki točki učenja se začne mreža prilagajati tem značilnostim učnih podatkov in tako se začne napaka na testnih podatkih višati, čeprav se napaka na učnih podatkih še vedno manjša. To se zgodi, ko mrežo prenasičimo z znanjem (angl. *overtraining*). Poleg tega se lahko tako zelo prilagodi podatkom, da se odziva le na en tip podatkov, to je na učno množico. Tako že čisto majhne spremembe na vhodu vplivajo na napačno klasifikacijo. Te lastnosti pomenijo, da se dobro prilega učnim primerom, ima pa slabe generalizacijske zmožnosti. Generalizacija je tako zaželena in kritična lastnost.

Eno od odprtih vprašanj torej vključuje določanje najbolj primerne velikosti nevronske mreže za reševanje določenih opravil, še posebno za njene praktične implementacije na tako pomembnih področjih, kot sta učenje in generalizacija. Glavno pravilo, ki se ga je potrebo držati, da ohranimo dobro generalizacijo v sistemu, ki je naučen na primerih, je, da uporabimo manjšo mrežo, ki se bo prilegala podatkom. Kar se tiče implementacije, so majhne nevronske mreže bolj prijazne, saj potrebujejo omejeno število virov v računskem okolju [7]. To je bolj zaželeno v praktični uporabi na manj zmogljivih računalnikih, saj porabijo manj računskega časa. V nekaterih primerih pa manjša nevronska mreža ni popolnoma imuna na lokalne minimume. Mreže z več skritimi enotami se medtem zmorejo naučiti hitro in se tudi lažje izognejo lokalnemu minimumu. Prevelika mreža pa bo nagnjena k pomnenju učnih vzorcev in bo zato imela slabše generalizacijske sposobnosti. Boljša predstavitev računske zmogljivosti se lahko doseže samo z majhnimi mrežami, vendar lahko za njihovo učenje porabimo veliko časa. Majhne mreže nimajo dovolj nevronov, da bi lahko nove lastnosti podatkovne zbirke preprosto predstavile na še nerabljenih nevronih, kot to lahko naredi globoka mreža. Navadna, manjša mreža z omejenimi nevroni se mora soočiti z bolj kompleksno predstavitvijo nevronov za vse možne lastnosti zbirke.

Torej tako velike kot tudi manjše mreže imajo svoje prednosti in slabosti. Z gradnjo velike mreže, ki jo porežemo, lažje dobimo optimizirano arhitekturo mreže. Poleg tega z metodo rezanja pridobimo več, kot pa če bi uporabili ravno prav veliko mrežo (ob predpostavki, da bi sploh poznali njeno velikost). Če poznamo optimalno velikost mreže, je najmanjša možna mreža ravno dovolj kompleksna, da ustreza podatkom, in je zato občutljiva na začetne pogoje in učne parametre [33]. Večje mreže se učijo hitreje in z manjšo občutljivostjo. Z rezanjem pa pripomoremo k izboljšani generalizaciji.

V procesu rezanja se izognemo tudi lokalnemu minimumu brez podaljšanja časa celotnega učenja. Kriterij, ki določi tiste parametre, ki se jih izloči iz mreže, igra pomembno vlogo v takšnih algoritmih. Nekateri algoritmi režejo uteži med nevronskimi nivoji, ostali režejo nevrone same. Poznamo tudi

pristope, ki so kombinacija obeh algoritmov.

2.3 Standardni pristopi k rezanju nevronskih mrež

Rezane nevronske mreže predstavljajo pomembno raziskovalno področje na območju strojnega učenja, sicer pa v aplikacijah še niso veliko uporabljene. Ker so nevronske mreže občutljive na podatkovne zbirke, se različni rezalni algoritmi lahko na določeno zbirko različno odzovejo. V nadaljevanju smo za primerjavo in analizo vzeli nekaj načinov rezanja mreže. Primerjali smo jih s predlaganim načinom rezanja, osnovanega na matrični faktorizaciji. Primerjavo smo izvedli na različnih zbirkah podatkov.

Najbolj osnovno rezanje je rezanje z metodo grobe sile (angl. *brute-force*), ki vsako utež postavi na 0 in vrednoti spremembe v napaki. Če se napaka zelo poveča, potem obnovi prejšnjo utež, sicer jo odstrani. Groba sila je morda najbolj pogost pristop za iskanje nevronske mreže z optimalno arhitekturo [1]. Temelji na nekaj uspešno naučenih manjših mrežah, dokler se ne najde najboljše in najmanjše topologije, ki še ustreza podatkom. Ta proces je časovno zelo potraten, saj vključuje učenje velikega števila mrež. Za mreže, ki so začetno majhne, tudi konvergenca ni vedno zagotovljena.

Ostale algoritme rezanja lahko razdelimo v dve večji skupini. Algoritmi prve skupine računajo občutljivost funkcijske napake v primeru odstranitve enot. Odstranijo tiste enote, ki najmanj vplivajo na povečanje napake na testni množici. Algoritmi druge skupine dodajo kriterijski funkciji pravila, s katerimi nagradi mrežo zaradi izbire dobre rešitve. Skupini se prekrivata, saj kriterijske funkcije lahko vključujejo tudi občutljivost funkcijske napake. Obstaja veliko rezalnih algoritmov za optimiziranje arhitekture nevronske mreže.

Metode, ki temeljijo na občutljivosti, naučeno mrežo spreminjajo. To dosežejo z ocenjevanjem občutljivosti na napako v primeru izključitve vsake uteži v mreži. Uteži ali nevrone, ki niso občutljivi na spremembo napake,

se lahko izloči po vsakem koraku učenja. Porezan sistem ni močno občutljiv na škodo, četudi ima manj parametrov. Varianca uteži v enotah je dober napovedovalec pomembnosti enote [33]. Pomembnost same enote je tudi dober napovedovalec za pričakovano povečanje relativne povprečne kvadratne napake (RMSE), če odstranimo enote z največjo utežjo [33]. Po učenju je za vsako utež poznana njena občutljivost. Tiste z najnižjo občutljivostjo, ki najmanj prispevajo oziroma imajo najmanjši vpliv na uspešnost mreže, lahko odstranimo. V primeru, da odstranimo vse povezave do neke enote, odstranimo tudi enoto samo. Enota brez vhodnih povezav daje konstanten izhod in se jo lahko odstrani potem, ko primerno prilagodimo naslednike. Največja slabost metod, ki računajo občutljivost, je, da ne zaznajo povezanih enot [33] in tako ne zagotavljajo, da bomo z njimi opazili vse pomembne procesne enote. Občutljivost je izračunana na eni sami uteži, saj se predpostavlja, da bomo odstranjevali le enoto ali utež. Po odstranitvni prve enote ostale občutljivosti mogoče niso povsem upravičene, predvsem pri manjših mrežah. Na primer, dve enoti skupaj nimata velikega vpliva na izhod, posamezna enota pa ga ima; v tem primeru nobena enota ne bo odstranjena. Če med vhodi obstajajo odvisnosti, analiza občutljivosti ni najbolj učinkovita. Delno povezane enote so manj vplivne, a bolj pogoste. Veliko algoritmov, ki temelji na analizi občutljivosti za optimizacijo nevronske mreže, je bilo predstavljenih s strani raziskovalcev [1]. Najbolj uveljavljena (angl. *state-of-the-art*) rezalna algoritma, ki temeljita na analizi občutljivosti, sta optimalna poškodba možganov (angl. *optimal brain damage* (OBD)) in optimalni možganski kirurg (angl. *optimal brain surgeon* (OBS)).

2.3.1 Optimal brain damage

Optimal brain damage [1, 5, 8, 22, 33] ocenjuje približek izstopajočih (angl. *saliency*) uteži (višek na učni množici), ki je rezultat odstranitve vsakega posameznega parametra v arhitekturi nevronske mreže z izračunom drugega odvoda na izhodni napaki nevronske mreže glede na dano posamezno utež. Metoda potem odstrani tisti parameter, ki najmanj izstopa (angl. *the lowest*

silency). Pri tej metodi se rezanje opravlja iterativno na dobro naučeni mreži na razumljivem nivoju. Metoda izračuna stopnjo izstopanja, izbriše uteži ali enote, ki ne izstopajo, in nadaljuje z novim učenjem in rezanjem. Metoda predvideva, da je funkcija napake kvadratna in da je Hessova matrika diagonalna. Kompleksnost mreže se zmanjša tudi z omejevanjem nekaterih uteži, da ostanejo enake. Slabost metode je, da zahteva shranjevanje Hessove matrike.

2.3.2 Optimal brain surgeon

Kmalu po razvitju OBD so z enako idejo razvili metodo *Optimal brain surgeon* (OBS) [1, 5, 8, 12, 33], ki prav tako temelji na izstopanju parametrov in je opisana s primerom učne funkcije s kvadratno napako. V primerjavi z OBD gre korak naprej; odstrani namreč diagonalno predpostavljeno idejo Hessove matrike, ki je uporabljena pri OBD. Diagonalna predpostavka je nepravilna in lahko vodi do odstranitve napačnih uteži. OBS nastavi inverzno Hessovo matriko za računanje optimalne uteži za rezanje, medtem ko sočasno rešuje optimizacijski problem nadgradnje uteži, potrebne za zmanjšanje napake mreže. Takšen pristop dovoli simultane nadgradnje vseh ostalih uteži, zato ponovno učenje ni potrebno. Sicer pa je ta metoda precej nepraktična za velike mreže [1]. Algoritem opazuje napako na validacijski množici in se konča, ko se napaka začne večati, vendar nima zagotovila, da učna krivulja prečka optimalno točko. Končne uteži so občutljive na učno dinamiko. Metoda OBS je bolj dostopna kot sta OBD in magnitudno osnovano rezanje (angl. *magnitude-based pruning* (MBP)), saj je OBS eden najbolj priljubljenih metod za rezanje [5, 8]. Tako kot OBD mora ohranjati Hessovo matriko, zato zahteva dodaten spomin. Danes poznamo mnogo algoritmov, ki temeljijo na metodah OBD in OBS.

2.3.3 Magnitudno osnovano rezanje

Preprost način rezanja mreže je z magnitudno osnovanim rezanjem (angl. *magnitude-based pruning* (MBP)) [1, 8], ki odstranjuje najmanjše uteži z mreže z domnevo, da so nepomembne. Rezalne metode, ki temeljijo na magnitudi, predvidevajo, da so majhne uteži najmanj občutljive in zato najmanj pomembne, zato režejo prav te. Velika pomanjkljivost teh metod je, da pogosto odstranijo pomembne poti v mreži. Razlog leži v tem, da lahko skupina povezanih uteži več manjših enot bolj pomembno vpliva na mrežo kot pa ena sama večja, kar pa te metode ne upoštevajo najbolje, saj gledajo na vsak nevron kot individualno neodvisno enoto.

2.3.4 Skeletonizacija

Znana metoda skeletonizacije (angl. *skeletonization* (SKL)) [27] uporablja znanje nevronske mreže, da določi funkcionalnost in ustreznost individualnih enot. Glavna ideja je, da se mreža iterativno uči do določene velikosti, nato pa se izračuna ustreznost uteži ali enot. Ta ustreznost potem določa, katere izmed uteži ali enot so najbolj kritične za uspešnost mreže. Temu sledi avtomatsko rezanje najmanj ustreznih enot, preko česar se zgradi skeletna verzija mreže. Ker se postopek skeletonizacije osredotoča na najbolj ustrezne enote, mora biti mera ustreznosti poznana. Ustreznostna metrika lahko prepozna, kateri vhodni ali skriti nevroni so najbolj kritični pri uspešnosti mreže. Metriki ustreznosti uspeva prepoznati tudi tiste skrite nivoje, ki so najbolj odgovorni za uspešnost mreže. To pomeni, da relativno pravilno odkriva najmanjšo množico vhodnih značilnosti, potrebnih za opis razreda.

2.3.5 Metode s kazensko dobo

Metode s kazensko dobo (angl. *penalty-term method*) [1, 5, 33] oziroma z razkrojem uteži dodajajo kazensko dobo kriterijski/stroškovni funkciji, ki jo želimo minimizirati. Manjše uteži se tako pri vzvratni propagaciji počasi prisili, da dobijo vrednost 0 in se jih tako odstrani (lahko že v času učenja).

Četudi uteži niso zares odstranjene, se mreža obnaša kot manjši sistem. Kazenske metode kaznujejo uteži na več različnih načinov in na ta način nevronsko mrežo v času učenja prisilijo, da se znebi nepotrebnih uteži.

2.3.6 Interaktivno rezanje

Pri interaktivnem rezanju [5, 33] graditelj mreže pregleduje mrežo, označuje tiste enote, ki so nepomembne in jih odstrani. Algoritem ima na voljo nekaj hevristik, ki določajo kandidate za odstranitev. Recimo, če ima enota konstanten izhod skozi celotno učenje, potem ne sodeluje pri učenju, zato jo lahko odstranimo. Ker pa lahko prireja pristranskost nevronov v naslednjih nivojih, je treba njihove pragove (angl. *tresholds*) prilagoditi. Če ima neko število enot visoko povezan odziv (isti ali nasproten) skozi vse vzorce, potem so odvečne in se jih lahko združi v eno enoto oziroma nevron. Tudi njihove izhodne uteži se doda skupaj, tako da ima skupna enota enak učinek na naslednje.

2.3.7 Evolucijske rezalne metode

Znane in uspešne so tudi evolucijske rezalne metode [1, 10, 33]. Te uporabljajo genetski algoritem (angl. g*enetic algorithm*) za rezanje nevronskih mrež. Različne rezane mreže so ustvarjene preko mutacij, reprodukcij in križanj. Vsak posameznik v populaciji predstavlja eno verzijo rezane mreže. Po parjenju potomci predstavljajo nevronske mreže, ki so drugače porezane. Mreže so ocenjene glede na napako, ki jo dosežejo po učenju. Večje možnosti za reprodukcijo imajo mreže, ki uporabljajo manj parametrov in izboljšajo generalizacijo. Te metode dobro delujejo na manjših mrežah.

2.3.8 Lokalni in razporejeni zastoji

Prepoznavnost sta pridobili tudi metodi lokalni zastoji (angl. *local bottlenecks*) in razporejeni zastoji (angl. *distributed bottlenecks*) istega avtorja [5, 33, 40]. Pri prvi metodi skrite enote tekmujejo ena z drugo za preživetje, pri

čemer se računa prispevek posamezne enote k funkciji, izračunani v mreži. Prispevek se izračuna na podlagi razpršenosti vektorja uteži enote. Tisti nevroni, ki ne sodelujejo pri določenih opravilih, so torej odstranjeni. Druga metoda pa namesto rezanja uteži deluje tako, da na uteži postavi omejitve. Na ta način pomaga pri redukciji dimenzionalnosti.

2.4 Aproksimacija matrik z matrično faktorizacijo

Za boljše obvladovanje algoritmov podatkovne analize pri iskanju atributov v podatkih ali iskanju latentnih spremenljivk, se lahko uporabi zmanjšanje dimenzije. Skrčen model zagotavlja informacije skoraj na nivoju izvirnega sistema. Najbolj preprosta oblika za zmanjšanje dimenzije je matrična faktorizacija.

Izrek 2.1 *Dana je matrika* $A \in \Re^{m \times n}$ *in pozitivno celo število* $k<min\{m,n\}$. *Najdi matriki* $W \in \Re^{m \times k}$ *in* $H \in \Re^{k \times n}$, *kjer je* $WH \approx A$. *Produkt WH se imenuje matrična faktorizacija (MF) matrike A. WH je aproksimirana faktorizacija ranga* $r = k$.

Matrična faktorizacija je tehnika za iskanje linearne predstavitve, ki temelji na razcepljanju (faktoriziranju) le-teh. Kot smo omenili, je zbirka slik pogosto predstavljena v obliki matrik, kjer vsak stolpec predstavlja eno sliko znotraj zbirke. Matrična faktorizacija nato ustvari slovar s k baznimi slikami, ki so predstavljene v stolpcih faktorske matrike W. V matriki H pa najdemo koeficiente linearne kombinacije teh slik za predstavitev približka vsake od izvornih slik (matrike A) v zbirki [6].

Predpostavimo, da ima W veliko manj stolpcev, kot ima A vrstic, tako da je aproksimacija uspešna samo, če odkrije latentno strukturo v podatkih. Uporabna predstavitev ponavadi pokaže sicer skrite (latentne) strukture in pogosto zmanjša dimenzionalnost podatkov.

Aplikacije, kot so obdelava besedil, slik in podatkovno rudarjenje, ohranjajo koristne informacije v zelo velikih matrikah. Poleg tega, da matrika nizkega ranga porabi manj spomina, ponuja tudi bolj jasno in učinkovito predstavitev povezav med elementi podatkov [28]. Aproksimacija nizkega ranga najde ključne komponente podatkov in ne upošteva neključne komponente, ki jih pripisujemo šumu, napaki ali neskladnosti [19].

Zaradi potenciala, ki ga nosi matrična faktorizacija, se je razvilo veliko različic le-te. Zelo znana je nenegativna matrična faktorizacija (NMF), ki predpostavlja nenegativnost v podatkih, in posledično tudi v faktoriziranem modelu [4, 16, 23, 24, 25, 35]. Ta bi bila za naš primer problematična, saj uporabljamo negativne uteži. Najbolj standardna metoda za zmanjšanje dimenzije na podlagi matrične faktorizacije je analiza glavnih komponent (angl. *principal component analysis* (PCA)) [38], ki izračuna množico lastnih vektorjev s faktorizacijo, imenovano razcep s singularnimi vrednostmi (angl. *singular value decomposition* (SVD)). SVD je klasična tehnika linearne algebre, ki v kombinaciji z metodo najmanjših kvadratov uporabi lastne vektorje za optimalno približanje oziroma aproksimacijo visoko-dimenzionalnim podatkom. Ker je lastnih vektorjev manj kot je dimenzij podatkov, je končni prikaz podatkov veliko manjše dimenzije. PCA je preprosta za računanje in deluje dobro, vendar SVD uporabnikom ne ponuja interpertacij za matematične faktorje ali zakaj tako dobro deluje [19].

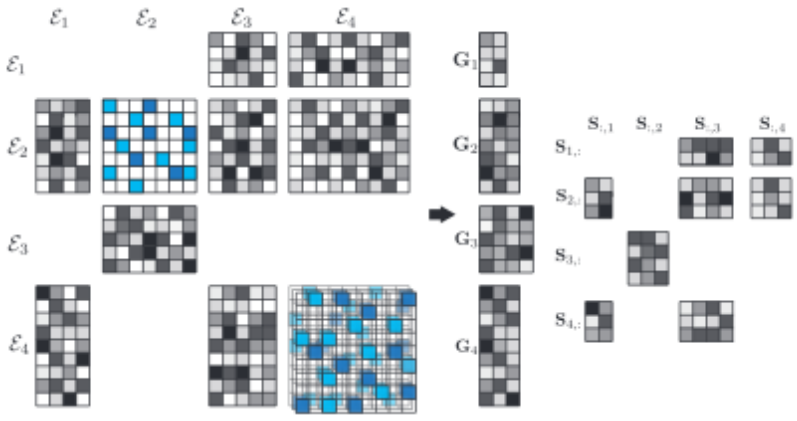
V našem primeru imamo opravka z več matrikami, nad katerimi moramo izvesti matrično faktorizacijo. Zato smo vzeli nadgrajeno obliko matrične faktorizacije, in sicer hkratno tri-faktorizacijo več povezanih matrik oziroma zlitje podatkov (angl. *data fusion*) [43].

Izrek 2.2 *Hkratna tri-faktorizacija več matrik hkrati faktorizira vse razpoložljive relacijske matrike R_{ij} v $G_i \in \Re^{n_i \times k_i}$, $G_j \in \Re^{n_j \times k_j}$ in $S \in \Re^{k_i \times k_j}$ in regularizira njihov približek skozi omejitvene matrike θ_i in θ_j, tako da velja $R_{ij} \approx G_i S_{ij} G_j^T$ [43].*

Kaznovana matrična tri-faktorizacija deluje tako, da se osredotoči na določeno ciljno relacijo in izkoristi neposredno povezane podatke (slika 2.3). Vsebuje

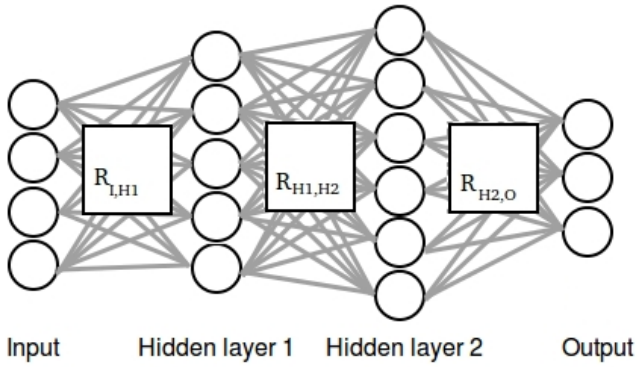
množico omejitev tistih povezav, ki se morajo povezati, in tistih, ki se ne smejo; tako vključuje relacije med objekti istega tipa [42]. Kaznovana matrična tri-faktorizacija simultano faktorizira matrike in odkriva skrite asociacije med matrikami. Ta pristop neposredno upošteva vsako podatkovno zbirko, ki jo lahko predstavimo z matriko. Algoritem predvideva, da so matrike podatkov, predvidene za zlitje, paroma povezane. To je, dve matriki si delita enak tip objektov ε (skupna dimenzija).

Podatkovna struktura, ki tako zagotavlja relacije med matrikami istega tipa ε_i, je predstavljena z matriko omejitev $\theta \in \Re^{n_i \times n_i}$. Četudi niso na voljo popolnoma vsi pari tipov objektov, je ta algoritem zmožen integrirati vse dostopne podatke, v primeru, da je glavni graf relacij med objekti povezan. Da ohranimo bločno strukturo našega sistema, je predlagana uporaba simultane faktorizacije vseh relacij matrik R omejenih z matriko θ. Nastali sistem vsebuje faktorje, ki so specifični za vsak podatkovni vir in faktorje, ki so specifični za vsak objektivni tip. Z deljenjem faktorjev dosežemo zlitje podatkov in hkrati identificiramo vzorce, značilne za posamezne vire [43].



Slika 2.3: Primer zlitja s hkratno tri-faktorizacijo več matrik. Na levi strani imamo vse razpoložljive relacijske matrike R, katere algoritem hkrati faktorizira v matrike G in S (na desni), ki tako hranijo razcepne faktorje [41]. Vir slike [43].

2.5 Aproksimacija in rezanje nevronskih mrež z matrično faktorizacijo

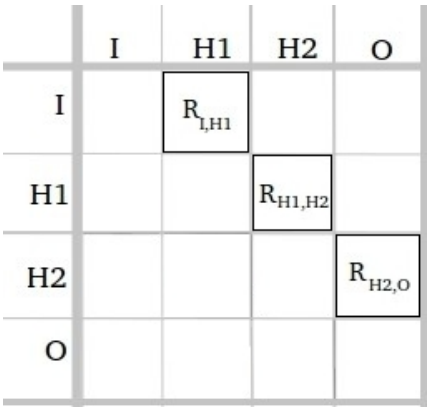


Slika 2.4: Nevronska mreža in matrike uteži povezav (relacijske matrike R) med nivoji. Velikost posamezne matrike je odvisna od števila enot na nivoju pred in za njo. Na primer, $R_{I,H1}$ ima 4 (I) vrstice in 5 ($H1$) stolpcev.

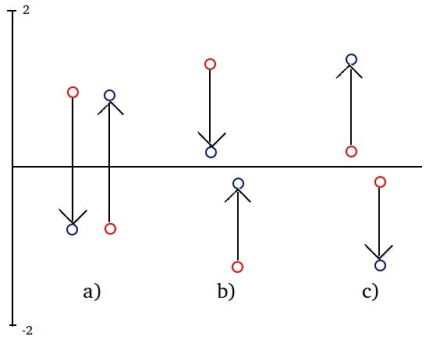
S kaznovano matrično tri-faktrizacijo smo rešili tudi problem globokih nevronskih mrež, in sicer večnivojsko arhitekturo z deljenimi utežmi. Pri navadnih nevronskih mrežah, kjer imamo samo en skriti nivo, najdemo dve matriki uteži, ki imata skupni objektni tip (skupno dimenzijo).

Zaradi te lastnosti ju je mogoče, preko te skupne dimenzije, združiti v eno matriko in na njej izvesti matrično faktorizacijo. V globokih mrežah imajo prav tako sosednje uteži skupni tip objektov (prikazano na slikah 2.4 in 2.5), tako da je možno pokazati soodvisnost med sosednjima matrikama uteži. Ni pa možno uporabiti soodvisnosti med, na primer, prvo in tretjo matriko uteži. S hkratno tri-faktorizacijo več matrik smo ta problem rešili in rezanje nevronskih mrež poimenovali rezanje možganov z matrično faktorizacijo (angl. *matrix factorization-based brain pruning* (MFBP)).

Uteži v naučeni nevronski mreži smo predstavili z zbirko matrik, nad katerimi smo izvedli hkratno tri-faktorizacijo. Končni rezultat je bil približek



Slika 2.5: Konfiguracija relacijskih matrik uteži R iz slike 2.4 za kaznovano matrično tri-faktorizacijo. V našem primeru relacijske matrike R predstavljajo matrike uteži nevronskih mrež. Konfiguracija poteka vzporedno z diagonalo, saj imajo sosednje matrike uteži skupno dimenzijo zaradi skupnega skritega nivoja, ki je med njima (opazimo na sliki 2.4) z določenim številom enot.



Slika 2.6: Možne spremembe uteži povezav po aproksimaciji z matrično faktorizacijo. Rdeč krogec predstavlja vrednost uteži pred uporabo matrične faktorizacije in moder aproksimirano vrednost uteži: a) uteži, katerim se predznak spremeni, b) uteži, ki se približajo vrednosti 0 in c) uteži, ki se oddaljijo od 0.

matrike uteži, kar smo nadalje uporabili za določanje porezanih uteži. V teoriji velja, da je neka utež porezana, ko je njena vrednost enaka 0. Ker pa so v praksi uteži skoraj vedno neničelne, je bilo potrebno oceniti, katere

uteži rezati.

Na sliki 2.6 so prikazane možne spremembe uteži od prvotno naučenih vrednosti in vrednosti, ki jih dobimo z aproksimacijo na osnovi matrične faktorizacije. Uteži se lahko spremeni predznak (a), lahko se približa ničli (b) ali pa se od nje oddalji (c). Predvidevamo, da so uteži tipa (b) nepomembne, medtem ko so spremembe uteži tipa (c) pomembne. Pomembnost sprememb tipa (a), t.j. sprememba predznaka uteži, je vprašljiva, saj tovrstne spremembe uteži bistveno spremenijo izračune v enoti.

2.5.1 Prvi kriterij - prevelike spremembe

Pri določanju kandidatov za rezanje smo uporabili dva kriterija. Slika 2.6(a-c) prikazuje prvi kriterij (MFBP1). Kriterij ohrani tiste uteži, ki se po uporabi aproksimacije z matrično faktorizacijo malo spremenijo. Odstranimo torej uteži, katerih vrednosti se bistveno spremenijo, kar zapišemo:

```
firstCriterium = abs(weight_matrix - weight_matrixMF)
nonCandidates = firstCriterium < 0,1
weight_matrix = weight_matrix * nonCandidates
```

Matrika $weight\_matrix$ vsebuje začetne vrednosti, matrika $weight\_matrixMF$ vsebuje aproksimirane vrednosti uteži. Uteži, katerim se absolutna vrednost ne spremeni bistveno (manj kot za 0,1), so v matriki $nonCandidates$ označene z 1; le-te ohranimo, ostale pa porežemo (postavimo na 0).

2.5.2 Drugi kriterij - približevanje ničli

Drugi kriterij (MFBP2) obravnava le spremembe uteži, prikazane na sliki 2.6c. Kriterij ohrani tiste uteži, ki se ne približajo vrednosti 0. Odstranijo pa se uteži, katerih absolutna vrednost se bistveno približa 0 (sprememba proti 0 več od 0,1), kar zapišemo kot:

```
secondCriterium = abs(weight_matrix) - abs(weight_matrixMF)
nonCandidates = secondCriterium < 0,1
weight_matrix = weight_matrix * nonCandidates
```

Izbrane uteži postavimo na 0 in jih na tej vrednosti zadržimo tudi ob ponovnem učenju in popravljanju.

Ker z drugim kriterijem režemo samo povezave, ki se po aproksimaciji približajo 0, kot je prikazano na sliki 2.6b, predvidevamo, da se bo bolje izkazal. Prvi kriterij rezanja lahko, poleg nepomembnih povezav (b), reže še vprašljive spremembe (a) kot tudi pomembne povezave (c).

2.5.3 Predlagani algoritem za rezanje nevronskih mrež

Prikazan je celotni algoritem *rezanje možganov z matrično faktorizacijo* (angl. *matrix factorization-based brain pruning* (MFBP)), ki vključuje uporabo matrične tri-faktorizacije za rezanje povezav:

Rezanje možganov z matrično faktorizacijo (MFBP)

vhod: nevronska mreža, shranjena v hiddenLayer[]

izhod: porezana nevronska mreža, shranjena v hiddenLayer[]

 # Nevronsko mrežo pretvori v množico matrik za faktorizacijo.

for $i := 0$ to $numOfWMatrices$ do

$weightMatrix[i] = hiddenLayer[i].W.getValue();$ od

for $i := 0$ to $numOfWMatrices$ step 2 do

$t[i] = fusion.ObjectType('Type' + str(i), rank)$

$t[i+1] = fusion.ObjectType('Type' + str(i+1), rank)$

$relations.add(fusion.Relation(weightMatrix[i/2], t[i], t[i+1]))$ od

$fusionGraph = fusion.FusionGraph()$

$fusionGraph.add\_relations\_from(relations)$

$fuser = fusion.Dfmf()$

 # Izvedi faktorizacijo.

$fuser.fuse(fusionGraph)$

 # Primerjaj izvorne in aproksimirane uteži ter poreži.

$nonCandidates = []$

for $i := 0$ to $numOfWMatrices1$ do

$weightMatrixMF = fuser.complete(relations[i])$

$diff = abs(weightMatrix[i] - weightMatrixMF)$ **# za MFBP1**

$diff = abs(weightMatrix[i]) - abs(weightMatrixMF)$ **# za MFBP2**

$nonCandidates.append(diff < 0.1)$

$weightMatrixMF = weightMatrix[i] * noncandidates[i]$

$hiddenLayer[i].W.setValue(weightMatrixMF)$ od

 # Izboljšaj neničelne uteži s ponovno iteracijo učenja.

$retrain\_one\_epoch(hiddenLayer[])$

 # Poskrbi, da imajo porezane povezave utež 0.

for $i := 0$ to $numOfWMatrices$ do

$weightMatrix = hiddenLayer[i].W.getValue()$

$weightMatrixMF = weightMatrix * nonCandidates[i]$

$hiddenLayer[i].W.setValue(weightMatrixMF)$ od

return hiddenLayer

2.6 Postopek gradnje in rezanja

Za ustvarjanje nevronske mreže smo uporabili programski jezik Python in knjižnico scikit-fusion [43] za hkratno faktorizacijo več matrik (zlivanje podatkov), ki temelji na tri-faktorizaciji in nedavno razvitih kolektivnih latentnih faktorskih modelih.

Za delo z nevronskimi mrežami v Pythonu smo poiskali knjižnico Theano [3], ki je primerna predvsem za gradnjo nevronskih mrež. Daje popoln nadzor nad snovanjem nevronske mreže, kar je bilo v našem primeru potrebno zaradi prilagajanja uteži povezav. Uporabili smo na Theanu zgrajeno knjižnico PyLearn2 [11], ki vsebuje implementacije mnogih algoritmov za uporabo nevronskih mrež in strojnega učenja. S PyLearn2 smo lahko uporabili prednosti znane knjižnice za strojno učenje Scikit-learn [30].

Mreži je bilo potrebno določiti tudi število skritih nevronov, globino mreže in ustavljalni kriterij. Pri tem smo upoštevali, da morajo biti parametri takšni, da bodo ustrezni za različne zbirke podatkov.

Za število skritih enot smo se zgledovali po viru [18]. Vsi skriti nivoji imajo enako število skritih enot, vsak skriti nivo pa vsebuje takšno število skritih nevronov, ki ustreza vsoti vhodnih in izhodnih nevronov. Če pa je skritih nevronov več kot 100, to vsoto množimo z 2/3.

```
num_hidden = num_inputs + num_outputs
if num_hidden > 100:
    num_hidden = num_hidden * 2 / 3
```

Primerno zgrajeni nevronski mreži smo vzeli matrike uteži, nad katerimi smo izvedli matrično tri-faktorizacijo. Uporabili smo navadno in globoko mrežo, pri čemer je navadna mreža vsebovala en skriti nivo, globoka pa pet.

Za analizo uspešnosti rezalnega algoritma z matrično faktorizacijo je bila narejena primerjava z nekaterimi standardnimi algoritmi. Ker na spletu ni bilo mogoče najti že spisane odprte kode katerega od standardnih rezalnih algoritmov, smo pri programskemu jeziku R uporabili knjižnico RSNNS (*Stuttgart neural network simulator* (SNNS) za R). RSNNS je knjižnica, ki vse-

buje veliko standardnih implementacij za nevronske mreže. Omogoča vse algoritmične funkcionalnosti SNNS znotraj R. Vsebuje tudi nekaj rezalnih algoritmov, od katerih smo uporabili MBP, OBD, OBS in SKL.

2.7 Vrednotenje napovednih modelov

Za vrednotenje napovedne uspešnosti nevronskih mrež smo uporabili metodo prečnega preverjanja. Celotna zbirka je razdeljena na učno, testno in validacijsko množico. Če nevronska mreža z učenjem izboljša uspešnost na validacijski množici, se učenje nadaljuje, sicer se ustavi.

Ker pri metodah občutljivosti ne vemo, kdaj končati z rezanjem, lahko tudi tu uporabimo metodo prečnega preverjanja na validacijski množici, kjer je poleg rezalnega kriterija vključen validacijski korak za testiranje rezane mreže. Rezanje se ustavi, ko občutljivost na validacijski množici nenadoma skoči. Opisano prečno preverjanje smo uporabili pri rezanju s standardnimi algoritmi.

2.7.1 Prečno preverjanje

Za prečno preverjanje smo uporabili metodo razslojenega K-kratnega prečnega preverjanja (angl. *K-fold stratified*) brez ponavljanj, kjer je K enak 10. Desetkratno prečno preverjanje primere v podatkovni zbirki naključno razdeli v deset skupin. Zaradi lastnosti razslojevanja primere razdeli v skupine tako, da je razmerje pripadnikov posameznega razreda enako razmerju v vseh podatkih. Metoda v vsaki od deset iteracij eno skupino vzame za testno in validacijsko množico, ostale pa za učno množico. Tovrstno prečno preverjanje boljše oceni varianco pri enem poganjanju oziroma pri eni testni množici, kar je velikega pomena, če je količina podatkov v zbirki omejena.

2.7.2 Nastavljanje parametrov z uporabo internega prečnega preverjanja

Znano je, da če hočemo z matrično faktorizacijo zmanjšati dimenzijo uteži povezav, mora biti rang (strogo) manjši od dimenzij izvorne matrike. V ta namen je bilo uporabljeno interno prečno preverjanje, kjer zgrajeno mrežo režemo z matrično faktorizacijo različnih rangov. Algoritem na zgrajeni mreži iterira in nad njenimi utežmi računa matrično faktorizacijo, najprej z $rang = 2$ in vse do največjega dovoljenega ranga, to je ranga, ki je za eno manjši od najmanjše dimenzije matrike uteži. Pri vsaki porezani mreži različnega ranga se izračunajo uspešnost z AUC (angl. *area under curve*), aproksimacija matrike, ki je rezultat matrične faktorizacije z izvorno matriko, in delež porezanih uteži.

2.7.3 Računanje napovedne točnosti

Napovedno uspešnost smo merili s površino pod krivuljo AUC, kjer je krivulja ROC (angl. *receiver operating characteristics*). Krivulja ROC binarnega klasifikatorja je grafična predstavitev kompromisa med verjetnostjo, da imamo resničen pozitiven primer oziroma občutljivost (angl. *sensitivity*) na abscisni osi, in verjetnostjo napačno pozitivnih primerov na ordinatni osi. Uspešnost testne množice je izmerjena s površino pod krivuljo ROC, torej z AUC. Popolno uspešnost napovedi predstavlja $AUC = 1$, medtem ko $AUC = 0.5$ predstavlja naključno ugibanje. Bližje, kot je krivulja zgornji levi meji, bolje ločuje med razredi. Velja tudi, da je delež pravilno pozitivno napovedanih (angl. *true positive rate*) visok ter delež napačno pozitivno napovedanih (angl. *false positive rate*) nizek. Tako lahko površino pod ROC krivuljo uporabljamo kot merilo diskriminativnosti modela. Zaradi te značilnost se je ta krivulja začela vedno bolj pogosto uporabljati tudi pri strojnem učenju za ocenjevanje in primerjanje algoritmov. Razlog za vedno večjo uporabo takšnega načina je tudi v tem, da je preprosta klasifikacijska točnost pogosto prešibka metrika za merjenje uspešnosti [31]. Poleg tega, da je splošno

uporabna grafična metoda, ima lastnosti, zaradi katerih je še posebej uporabna za domene z neuravnoteženo distribucijo razredov in neenakimi cenami klasifikacijskih napak. Te lastnosti postajajo vse bolj pomembne, saj raziskave še naprej vztrajajo na območjih cenovno občutljivega učenja in učenja z neuravnoteženimi razredi.

Če je možnih razredov več kot dva, imamo večrazredni problem in moramo AUC računati drugače. Načinov reševanja takšnega problema je več in nosijo ime večrazredni AUC (angl. *multi-class AUC*). Uporablili smo enega bolj pogostih načinov računanja, to je ROC eden proti vsem. Tu večrazredni klasifikator predstavimo kot binarni problem. Iteriramo skozi razrede, kjer dobljeni razred uporabimo kot pozitivnega, vse ostale pa kot negativne. Nato izračunamo krivuljo ROC in njeno AUC. Pri tem nastane toliko rezultatov uspešnosti AUC, kolikor je razredov, zato jih povprečimo v eno oceno.

2.8 Opis podatkovnih zbirk za vrednotenje učnih algoritmov

Podatkovne zbirke za delo z nevronskimi mrežami so bile zbrane iz vira [26]. Izbrali smo predvsem večje zbirke s 5000 do približno 1.000.000 primeri. Tudi atributi so bili različnega tipa in so variirali od devet do 170 možnih atributov. Na raznolikih zbirkah smo želeli izvedeti, kako bosta potekali gradnja nevronske mreže in njeno rezanje. Vse uporabljene zbirke so namenjene predvsem reševanju klasifikacijskih primerov. Ker nekatere zbirke vsebujejo vrednosti, ki jih algoritem ne sprejme, so bili ti atributi pretvorjeni v numerični format.

2.8.1 Wave

Zbirka *Waveform Database Generator* iz generatorja zbirke v valovni obliki po knjigi CART.

Št. razredov	Št. primerov	Št. atributov	Št. manjkajočih vrednosti
3	5.000	40	Brez

Vir: Breiman,L., Friedman,J.H., Olshen,R.A., Stone,C.J. (1984), pripravil David Aha.

Zbirka ima tri razrede valov in 40 atributov, pri čemer vsi vsebujejo šum. Zadnjih 19 atributov je šumnih atributov s povprečjem 0 in varianco 1. Vsak razred je generiran iz kombinacije dveh od treh možnih baznih valov. Vsaka instanca je generirana z dodanim šumom v vsakem atributu. Za poenostavitev imena se v nadaljevanju zbirka imenuje Wave.

2.8.2 Nursery

Podatkovna zbirka Nursery je bila izpeljana iz modela hierarhičnega odločanja in prvotno razvita za razvrščanje vlog za v vrtce.

Št. razredov	Št. primerov	Št. atributov	Št. manjkajočih vrednosti
4	12.960	8	Brez

Vir: Vladislav Rajkovic et al. (13 experts), pripravila Marko Bohanec, Blaz Zupan.

Podatkovna zbirka je bila zbrana v osemdesetih letih prejšnjega stoletja, ko so bili vpisi v vrtce v Ljubljani izredno veliki. Ker nekateri otroci tako niso bili sprejeti v vrtec, so bila pogosto potrebna objektivna pojasnila. Končna odločitev je bila odvisna od treh podproblemov: lokacije staršev in vrtca, družinske strukture in finančne ravni družine ter družbene in zdravstvene slike družine.

2.8.3 Shuttle

Podatkovna zbirka *Statlog (Shuttle)* vsebuje devet numeričnih atributov, pri čemer jih približno 80 % ustreza prvemu razredu.

Št. razredov	Št. primerov	Št. atributov	Št. manjkajočih vrednosti
7	58.000	9	Brez

Vir: Jason Catlett, Basser Department of Computer Science, University of Sydney, Australia.

Približno 80 % podatkov pripada prvemu razredu, kar je tudi privzeta točnost. Primeri so bili v izvorni zbirki v časovnem zaporedju, kar je relevantno pri klasifikaciji. Ker to ni bilo pomembno pri nalogah, namenjenih za delo na tej zbirki, se je naredila naključna razporeditev primerov. Za poenostavitev imena se v nadaljevanju zbirka imenuje Shuttle.

2.8.4 Clean (Musk)

Podatkovna zbirka *Musk* je bila namenjena napovednemu modelu, ki je napovedoval, ali so molekule dišeče (mošusne) ali ne.

Št. razredov	Št. primerov	Št. atributov	Št. manjkajočih vrednosti
2	7.074	168	Brez

Vir: AI Group at Arris Pharmaceutical Corporation, pripravil Tom Dietterich, Department of Computer Science, Oregon State University.

Podatkovna baza je skupek dveh zbirk: mošusa verzije 1 in 2. Prva zbirka vsebuje 6598 primerov, druga zbirka pa preostalih 476 primerov. Prva vsebuje množico 92 molekul (atributov), od katerih so jih poznavalci 47 predlagali za mošusne. Druga vsebuje 102 molekul, kjer je bilo 39 od njih ocenjenih, da so mošusne. Cilj je bil naučiti se napovedovati, ali bodo nove molekule mošusne ali ne. Za generacijo te zbirke je bilo uporabljeno nizkoenergijsko razporejanje molekul, čemur je sledila filtracija z odstranjevanjem najbolj podobnih razporeditev. Nato je bil določen vektor atributov, ki je opisoval vsako razporeditev. Za poenostavitev imena zbirko v nadaljevanju imenujemo Clean.

2.8.5 Diagnosis

Podatkovna zbirka *Dataset for sensorless Drive Diagnosis* opisuje tok motorja in diagnoze vožnje brez senzorjev. Motor ima tako nedotaknjene kot poškodovane dele, kar posledično prispeva k 11 različnim razredom z različnimi pogoji.

Št. razredov	Št. primerov	Št. atributov	Št. manjkajočih vrednosti
11	58.509	49	Brez

Vir: Martyna Bator (University of Applied Sciences, Ostwestfalen-Lippe), pripravil Martyna Bator.

Atributi so pridobljeni iz signalov električnega toka pogona. Pogon ima poškodovane in nepoškodovane dele, kar se pokaže v 11 različnih razredih z različnimi pogoji. Vsak pogoj je bil izmerjen večkrat z 12 različnimi operativnimi pogoji, kar pomeni povprečja pri različnih hitrostih, trenutkih zagonov in zagonskih silah. Signali tokov so bili izmerjeni s senzorjem toka in osciloskopom na dveh fazah. Za poenostavitev imena zbirko v nadaljevanju imenujemo Diagnosis.

2.8.6 Poker

Namen zbirke *Poker Hand* je bil napovedovanje rok pri pokru.

Št. razredov	Št. primerov	Št. atributov	Št. manjkajočih vrednosti
10	1.025.010	11	Brez

Vir: Robert Cattral, Franz Oppacher, Carleton University, Department of Computer Science, Ontario, Canada.

Vsak zapis je primer roke, ki vsebuje pet igralnih kart, vzetih iz standardnega kupa 52 kart. Vsaka karta je opisana z dvema atributoma (barva in rang) z vsemi desetimi napovednimi atributi. Atribut, ki predstavlja razred, opisuje končno kombinacijo kart (*Poker Hand*). Ker je pomemben tudi vrstni red kart, obstaja 480 možnih *Royal Flush-ov* v primerjavi s štirimi za vsako barvo. Ker bi dobrih milijon primerov za računanje mreže porabilo preveliko časa, smo zbirko najprej zmanjšali na 300.000 primerov v približno enakem razmerju razredov. Potem smo z bisekcijo poskusili odkriti še manjšo zbirko, katere uspešnost na mreži ni padla premočno. Za nadaljno delo smo vzeli vzorec 60.000 primerov. Za poenostavitev imena zbirko v nadaljevanju imenujemo Poker.

Poglavje 3

Rezultati

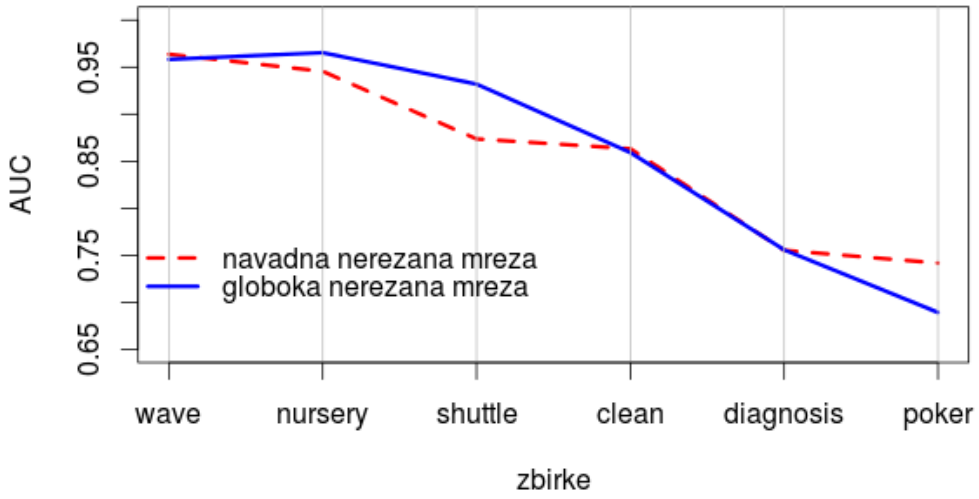
V tem poglavju sledijo odgovori na ciljna vprašanja, postavljena v diplomskem delu. Zanimalo nas je, kako se je odrezal rezalni algoritem z matrično faktorizacijo, katerim tipom nevronskih mrež (navadne ali globoke) rezanje bolj pripomore, kako se predlagani postopek rezanja primerja s standardnimi rezalnimi algoritmi ter kakšen je bil delež rezanja v primerjavi z napovedno uspešnostjo.

3.1 Napovedna uspešnost nerezane nevronske mreže

Uspešnost navadnih nerezanih mrež smo računali z večrazredno AUC. Rezultate prikazujeta tabela 3.1 in slika 3.1. Ti rezultati predstavljajo izhodiščne vrednosti napovedne uspešnosti, s katerimi smo primerjali uspešnost različnih pristopov rezanja.

	Wave	Nursery	Shuttle	Clean	Diagnosis	Poker
navadna mreža	0.964	0.946	0.874	0.863	0.755	0.742
globoka mreža	0.958	0.965	0.932	0.859	0.756	0.711

Tabela 3.1: Uspešnost (AUC) nerezanih nevronskih mrež.



Slika 3.1: Uspešnost (AUC) nerezanih nevronskih mrež.

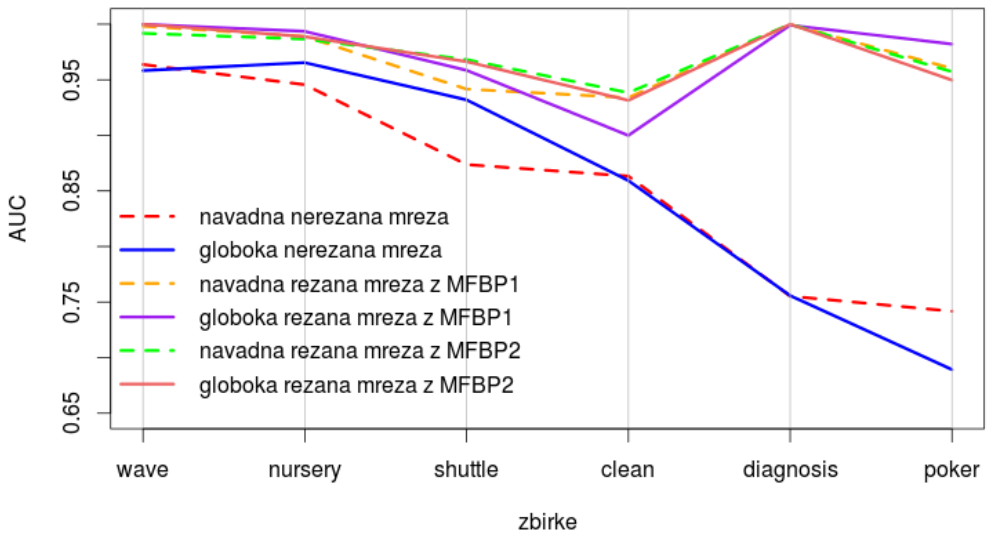
3.2 Napovedna uspešnost nevronskih mrež, porezanih z uporabo matrične faktorizacije

Rezanje nevronske mreže s predlagano metodo MFBP se je izvedlo na mreži z rezultati v tabeli 3.1. Predlagana metoda za izbiro porezanih uteži uporablja dva kriterija. Na porezanih mrežah je bila izmerjena uspešnost mreže z uporabo metode za ocenjevanje AUC na večrazrednih problemskih domenah. Na sliki 3.2 vidimo graf, ki prikazuje uspešnost rezalnega algoritma z matrično faktorizacijo v primerjavi z nerezano mrežo, medtem ko tabela 3.2 prikazuje uspešnost rezalnega algoritma z matrično faktrizacijo. Uspešnost nerezane mreže je prikazana v tabeli 3.1.

Rezanje je prikazalo izboljšanje uspešnosti mreže in s tem generalizacijske sposobnosti, zaradi česar lahko trdimo, da ima matrična faktorizacija potencial kot rezalni algoritem. Izboljšanje sta pokazali tako globoka kot navadna

		Wave	Nursery	Shuttle	Clean	Diagnosis	Poker
MFBP1	navadna mreža	0.998	0.989	0.942	0.934	0.9998	0.960
	globoka mreža	1.00	0.993	0.958	0.900	0.9993	0.982
MFBP2	navadna mreža	0.992	0.986	0.968	0.938	0.9999	0.957
	globoka mreža	1.00	0.989	0.966	0.928	0.999	0.950

Tabela 3.2: Uspešnost (AUC) mreže po rezanju z MFBP.



Slika 3.2: Uspešnost (AUC) mreže pred in po rezanju z MFBP.

mreža.

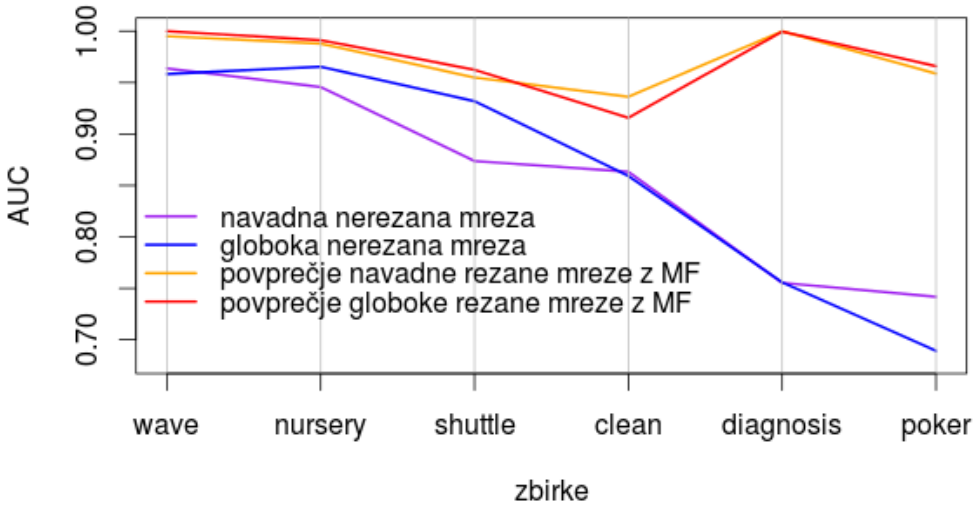
Pričakovano se je bolje odrezala matrična faktorizacija z drugim kriterijem rezanja, saj je povprečno dosegla AUC 0,9732, medtem ko je matrična faktorizacija s prvim kriterijem dosegla AUC 0,9713. Ker pa je odstopanja

le zanemarljivih AUC 0,0019, se moramo vprašati, zakaj je bil prvi kriterij tako dober, glede na to, da je morda rezal pomembne povezave. To bi bilo potrebno bolje preučiti, pri tem pa bi bilo še posebej zanimivo ugotoviti, kako na uspešnost vpliva rezanje tistih uteži, ki z matrično faktorizacijo spremenijo predznak (primer na sliki 2.6a).

3.3 Primerjava napovednih uspešnosti porezanih navadnih in globokih nevronskih mrež

Pričakujemo lahko, da bo rezanje najbolj koristilo globokim mrežam, saj je pri njih večja verjetnost, da smo začeli s preveliko mrežo. Vendar pa je to bolje, kot da začnemo s premajhno mrežo, četudi je lahko globoka že močno prevelika. Kdaj je mreža prevelika, je sicer težko ugotoviti. V nekaterih zbirkah se, na primer, lahko zgodi, da so že majhne mreže prevelike in tako primerne za rezanje. Z rezanjem pa lahko izboljšamo napovedno uspešnost na novih primerih tudi pri navadnih mrežah.

Na sliki 3.3 smo globoke in navadne mreže ločili z obliko narisane črte. Polne črte predstavljajo globoke mreže, medtem ko črtkane predstavljajo navadne. Na pogled je težko določiti, katere mreže so se bolje odrezale, zato lahko sklepamo, da je uspešnost rezanja na navadni in globoki podobno. Povprečno so globoke nerezane mreže dosegle AUC 0,85998, medtem ko so navadne dosegle AUC 0,85726. V nasprotju s tem so globoke rezane mreže dosegle izboljšanje na 0,97245, kar pomeni 0,11248 izboljšanja rezanja z matrično faktorizacijo. Navadne rezane mreže so dosegle AUC 0,97211, kar pomeni izboljšanje vrednosti za 0,11485. Rezanje navadnih mrež je pokazalo boljši rezultat za slabih 0,0024, kar je zanemarljivo malo. Sicer so globoke rezane mreže na vseh zbirkah, z izjemo zbirke Clean, dosegle boljše rezultate. Pri zbirki Shuttle je veliko večji uspeh dosegla navadna mreža, pri zbirki Poker pa globoka. Na globokih mrežah ne dosegamo boljših rezultatov oziroma



Slika 3.3: Uspešnost (AUC) navadnih in globokih mrež.

uspešnost matrične faktorizacije ni merljiva z globino mreže. Morda je razlog, zakaj se globoke mreže niso bolje izkazale, v tem, da parametri niso bili optimalno nastavljeni za globoke mreže. Ta problem bi bilo dobro bolje raziskati.

3.4 Primerjava z uspešnostjo standardnih pristopov rezanja

Na mrežah, ki so pred rezanjem dosegle uspešnost (AUC), ki jo prikazuje tabela 3.1, smo prav tako uporabili standardne rezalne algoritme. Uspešnost porezane mreže smo računali z večrazrednim AUC na navadnih (tabela 3.3) in globokih mrežah (tabela 3.4). Nekateri rezalni algoritmi so na določenih zbirkah potrebovali preveč časa za računanje, zato smo proces prekinili; v tabeli 3.4 zato nekateri rezultati niso podani. Primerjavo uspešnosti standardnih algoritmov z algoritmom matrične faktorizacije prvega in drugega

kriterija vidimo na sliki 3.4 na navadnih in na sliki 3.5 na globokih mrežah.

	Nursery	Shuttle	Poker
MBP	**0.994**	**0.980**	0.710
OBD	0.969	0.949	0.663
OBS	0.981	0.951	0.731
SKL	0.941	0.956	0.657
MFBP1	0.989	0.942	0.960
MFBP2	0.986	0.968	**0.975**

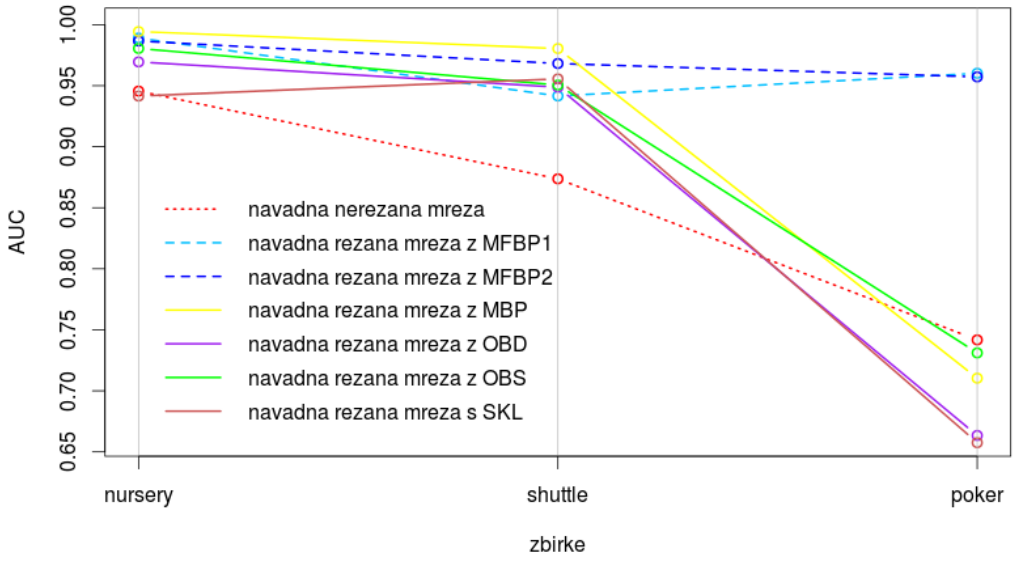
Tabela 3.3: Uspešnost (AUC) standardnih algoritmov na navadnih mrežah.

	Nursery	Shuttle	Poker
MBP	0.992	0.937	-
OBD	0.992	**0.969**	-
OBS	**0.995**	0.948	-
SKL	0.923	0.951	-
MFBP1	0.993	0.958	**0.982**
MFBP2	0.989	0.966	0.950

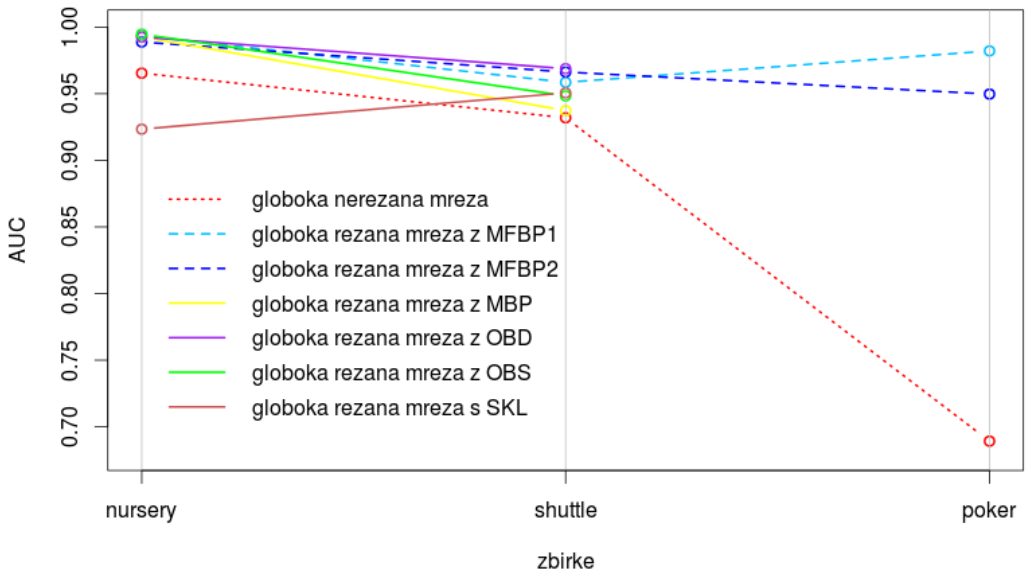
Tabela 3.4: Uspešnost (AUC) standardnih algoritmov na globokih mrežah.

Opozoriti moramo, da so standardni algoritmi imeli na voljo več iteracij rezanja, kjer so lahko po rezanju znova naučili mrežo, jo znova rezali in tako uspešnost mreže še izboljšali, dokler ni napaka rezane mreže dosegla vrednosti praga. Rezanje matrične faktorizacije se ni izvajalo na takšen način, torej pri tem rezanju ni bilo popravljanja in izpopolnjevanja mreže z večkratnim rezanjem. Algoritem matrične faktorizacije je izvedel samo eno iteracijo rezanja in nato še zadnje popravljanje mreže. V tej diplomski nalogi so tako rezultati predstavljeni izključno z uspešnostjo mreže po eni iteraciji rezanja.

Kljub temu da je algoritem rezanja z matrično faktorizacijo imel na voljo samo eno iteracijo rezanja, medtem ko so ostali algoritmi lahko popravljali večkrat, je matrična faktorizacija po uspešnosti primerljiva in v nekaterih primerih celo boljša v primerjavi z ostalimi standardnimi pristopi. Izboljšanje se opazi predvsem pri zbirki Poker. Težko pa bi trdili, da bi bili v primeru uporabe več iteracij rezanja rezultati uspešnosti še boljši, saj obstaja možnost, da bi naslednje rezanje že prečkalo prag. Zato je pomembno poudariti, da bi bilo to priporočljivo preveriti v nadaljnjih raziskavah.



Slika 3.4: Primerjava uspešnosti (AUC) rezalnega MFBP algoritma s standardnimi pristopi na navadni mreži.



Slika 3.5: Primerjava uspešnosti (AUC) rezalnega MFBP algoritma s standardnimi pristopi na globoki mreži.

3.5 Napovedna uspešnost v odvisnosti od stopnje rezanja

Zanimalo nas je tudi, kako na uspešnost vpliva stopnja oziroma delež rezanja. Predstavljeni algoritem MFBP uporablja dva kriterija rezanja, zato pričakujemo, da bo razmerje med napovedno uspešnostjo in deležem rezanja različno. V spodnji tabeli 3.5 so predstavljeni deleži rezanja na podlagi prvega in drugega kriterija rezanja.

		rang = 2		srednji rang		končni rang		
		navadna mreža	globoka mreža	navadna mreža	globoka mreža	navadna mreža	globoka mreža	Povprečni delež rezanja
MFBP1	posamičen	0.48	0.286	0.272	0.155	0.13	0.092	0.263
	povprečje	0.383		0.214		0.111		
MFBP2	posamičen	0.22	0.33	0.09	0.135	0.032	0.056	0.144
	povprečje	0.275		0.112		0.044		

Tabela 3.5: Posamezni in povprečni deleži rezanja navadne in globoke mreže s prvim in drugim kriterijem.

Iz tabele 3.5 lahko razberemo, da je bil pri najmanjšem rangu delež rezanja največji, pri največjem izračunanem rangu pa najmanjši. Več uteži je povprečno porezal prvi kriterij in to za približno 12 %. Predvidevamo, da je manjši delež rezanja pri drugem kriteriju posledica manjšega števila možnih kandidatov za rezanje. Rezultati so pokazali tudi razlike pri navadnih in globokih mrežah. Prvi kriterij je močneje rezal pri navadnih mrežah, drugi kriterij pa pri globokih.

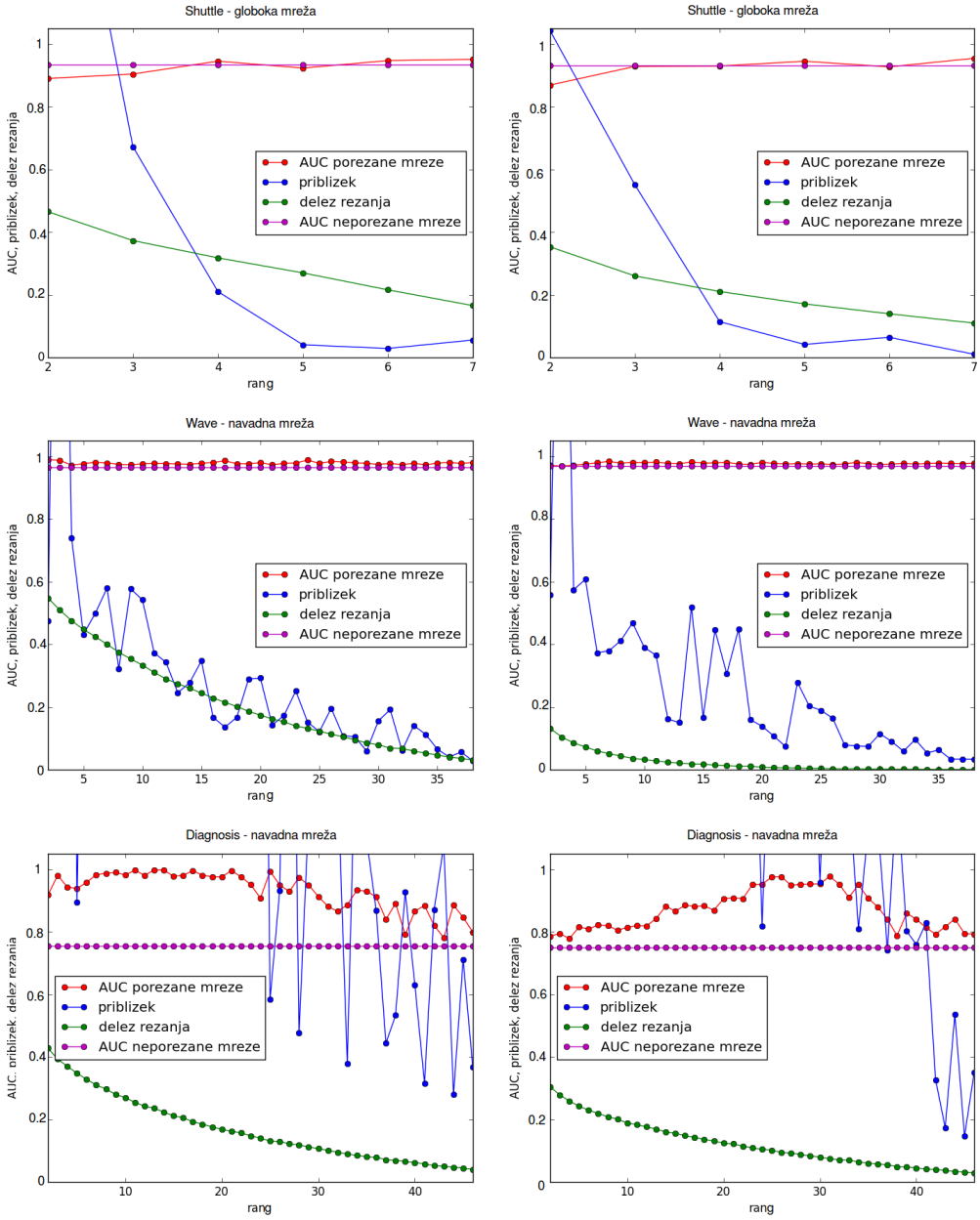
Pričakovali smo, da lahko rezanje do določene meje zaradi izboljšanja generalizacijske zmožnosti izboljša uspešnost nevronske mreže. A če gremo

z rezanjem predaleč, lahko začnemo rezati zelo pomembne povezave, kar rezultira v tem, da uspešnost mreže močno pade. V rezultatih nekaterih zbirk lahko pričakujemo, da bo pri najmanjših rangih rezanje premočno, kar bo vplivalo na padec uspešnosti. Nasprotno pa pri malo večjih zato pričakujemo izboljšanje. Lahko se zgodi tudi, da že pri največjem deležu rezanja (pri najmanjšem rangu) izboljšamo napovedno uspešnost. Na sliki 3.6 so rezultati predstavljeni s štirimi grafi, ki predstavljajo uspešnost porezane in nerezane mreže z AUC ter delež rezanja mreže. Slika prikazuje tudi približek porezane matrike uteži z nerezano, in sicer s kvadratno razliko matrik uteži nerezane nevronske mreže s porezano. Da bi bili rezultati čim bolj poenoteni, smo približek izračunali na sledeči način:

$$approx = \frac{(weight\_mat - weight\_matMF)^2}{weight\_mat^2} \tag{3.1}$$

Na grafih za posamezno zbirko ni bilo moč opaziti bistvene razlike med navadnimi in globokimi mrežami. Na sliki 3.6 smo prikazali različne možne izide in primerjali iste mreže porezane s prvim kriterijem rezanja na levi polovici slike z drugim kriterijem na desni polovici. Zbirka Shuttle prikazuje opisan primer slabše uspešnosti pri nižjih rangih v primerjavi z nerezanimi mrežami. Predvidevamo, da se je to zgodilo zaradi rezanja pomembnih povezav. Pri zbirki Diagnosis opazimo nasproten učinek, mreža je pri najmanjših rangih povprečno dosegala boljše rezultate. Drugačen vpliv je rezanje mreže imelo na zbirko Wave. Wave je tudi pri močnem rezanju prvih rangov (približno 55 % na navadni mreži, prvi kriterij) dosegal izboljšanje mreže. Višanje ranga matrične faktorizacije na zbirki Wave ni izboljšalo ali poslabšalo uspešnost napovedovanja.

Tudi kriterija sama sta na posamezni mreži prikazala podoben graf. Bistvena razlika rezanja med prvim in drugim kriterijem rezanja je zamik najvišje uspešnosti mreže na rangu. Mreža, porezana s prvim kriterijem matrične faktorizacije, je najvišjo uspešnost prikazala pri prvih oziroma na najnižjih rangih. Primeri rezultatov drugega kriterija pa so bili podobni rezultatom na zbirki Diagnosis, ki so dosegli najvišjo uspešnost nekje na sre-



Slika 3.6: Primeri rezanja s kriterijem MFBP1 (levo) in MFBP2 (desno).

dini možnih kandidatov rangov, t.j. med 25 in 35. Tudi na zbirki Shuttle je deloma viden zamik uspešnosti mreže. Tako lahko sklepamo, da je imel algoritem z drugim kriterijem rezanja v primerjavi s prvim pri enakem deležu

rezanja večjo verjetnost, da je porezal pomembne povezave.

Poglavje 4

Zaključek

Na nevronskih mrežah, zgrajenih iz šestih zbirk podatkov, smo uporabili dve različici predlaganega algoritma rezanja, ki temelji na hkratni faktorizaciji več matrik. Matrično faktorizacijo smo uporabili zaradi njene lastnosti aproksimacije v nižji dimenziji in odkrivanja latentnih povezav. Na nevronskih mrežah smo tako uporabili novo metodo zlitja podatkov z matrično tri-faktorizacijo [43]. Zanimalo nas je predvsem, kakšno napovedno uspešnost bomo dosegli po rezanju in koliko povezav bomo pri tem porezali. Za odločanje o rezanju posameznih povezav smo uporabili dva kriterija.

Ugotovili smo, da rezanje nevronske mreže z rezalnim algoritmom na podlagi hkratne faktorizacije več matrik izboljša uspešnost mreže. Rezanje na globoki in na navadni mreži je na vseh zbirkah izboljšalo uspešnost napovedovanja za povprečno 11,36 %. Največjo izboljšavo smo dosegli z rezanjem globoke mreže na zbirki Poker, in sicer za približno 27,67 %. Pri vsaki zbirki, pri obeh arhitekturah mreže (navadna in globoka), je uspešnost mreže presegala AUC 0,94. Presenetil nas je prvi kriterij rezanja z uspešnostjo (AUC) na ravni drugega kriterija. Pričakovali namreč smo, da se bo drugi kriterij zaradi njegove previdnosti izkazal veliko bolje, a to ni bilo tako. Vprašljive so bile uteži, ki so po matrični faktorizaciji spremenile predznak. Za nadaljnje delo se zdi smiselno bolj obširno raziskati, kako lahko spremembe uteži z matrično faktorizacijo vplivajo na uspešnost mreže, pri čemer se nanašamo

predvsem na spremembe tistih uteži, ki z matrično faktorizacijo spremenijo predznak.

Primerjava rezalnega algoritma s hkratno faktorizacijo več matrik in štirih standardnih pristopov rezanja je pokazala, da je predlagani algoritem rezanja po uspešnosti enakovreden ostalim algoritmom. Po uspešnosti je na vsaki zbirki eden od dveh kriterijev rezanja z matrično faktorizacijo dosegel vsaj drugo najvišjo uspešnost. Predvsem na zbirki Poker smo opazili znatno boljše napovedi v primerjavi z ostalimi rezalnimi algoritmi. Tu se je od standardnih algoritmov na navadni mreži najbolje izkazal OBS z AUC 0,731, ki pa je bil od našega algoritma s prvim kriterijem (AUC 0,96) slabši za 22,9 % in z drugim kriterijem (AUC 0,975) za 24,4 %. Zakaj so pri zbirki Poker standardni algoritmi pokazali slabše napovedi, bi si lahko razlagali s tem, da smo celotno zbirko Poker, ki presega milijon primerov podatkov, zredčili na 6 % (60.000 primerov). Pri nerezanih mrežah uspešnost napovedovanja ni bila bistveno manjša od zbirke s 30 % primeri, je pa lahko močno vplivala na uspešnost rezanja standardnih rezalnih algoritmov. V tem primeru se je naš rezalni algoritem z matrično faktorizacijo dobro izkazal, saj je lahko našel latentne povezave, ki so nadomestile pomanjkanje znanja. V prihodnje bi bilo dobro raziskati uspešnost rezalnega algoritma z matrično faktorizacijo na pomanjkljivih zbirkah in na zelo globokih mrežah, saj nam teh informacij ni uspelo dobiti.

Ker smo za primerjavo s standardnimi pristopi uporabili različna programska jezika in ne povsem enak način tretiranja mreže in rezanja, bi bilo v nadaljevanju potrebno bolj natančno raziskati primerljivost s standardnimi pristopi. Poleg uspešnosti, bi bilo smiselno primerjati še deleže rezanja na več zbirkah.

V diplomski nalogi smo uporabili gradnjo in rezanje na klasifikacijskih problemih. Dobro pa bi bilo preveriti tudi rezanje nevronske mreže na regresijskih problemih, saj je regresija pomembna problemska domena, kjer nevronske mreže pridobivajo vedno večjo pozornost.

Literatura

[1] M Gethsiyal Augasta and T Kathirvalavakumar. Pruning algorithms of neural networks—a comparative study. *Central European Journal of Computer Science*, 3(3):105–115, 2013.

[2] Dave Beeman. Neural network examples and demonstrations, 2000. Dosegljivo: `http://ecee.colorado.edu/~ecen4831/lectures/NNdemo.html`. [Dostopano 31.8.2015].

[3] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[4] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[5] Andrey Bondarenko and Arkady Borisov. Artificial neural network generalization and simplification via pruning. *Information Technology and Management Science*, 17(1):132–137, 2014.

[6] Christos Boutsidis and Efstratios Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[7] Giovanna Castellano, Anna Maria Fanelli, and Marcello Pelillo. An iterative pruning algorithm for feedforward neural networks. *Neural Networks, IEEE Transactions on*, 8(3):519–531, 1997.

[8] Giorgio Corani. Air quality prediction in milan: feed-forward neural networks, pruned neural networks and lazy learning. *Ecological Modelling*, 185(2):513–529, 2005.

[9] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[10] Rosa María García-Gimeno, César Hervás-Martínez, and Maria Isabel de Silóniz. Improving artificial neural networks with a pruning methodology and genetic algorithms for their application in microbial growth prediction in food. *International Journal of Food Microbiology*, 72(1):19–30, 2002.

[11] Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[12] Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

[13] Michael Heinert. Artificial neural networks–how to open the black boxes. *Application of Artificial Intelligence in Engineering Geodesy (AIEG 2008), S*, pages 42–62, 2008.

[14] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic mo-

deling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[15] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[16] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[17] Ehud D Karnin. A simple procedure for pruning back-propagation trained neural networks. *Neural Networks, IEEE Transactions on*, 1(2):239–242, 1990.

[18] Nate Kohl. Estimating the number of neurons and number of layers of an artificial neural network, 2010. Dosegljivo: `http://stackoverflow.com`. [Dostopano 31.8.2015].

[19] Amy N Langville, Carl D Meyer, Russell Albright, James Cox, and David Duling. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *arXiv preprint arXiv:1407.7299*, 2014.

[20] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40, 2009.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[22] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *NIPs*, volume 89, 1989.

[23] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[24] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[25] Tao Li and Chris Ding. The relationships among various nonnegative matrix factorization methods for clustering. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 362–371. IEEE, 2006.

[26] M. Lichman. Uci machine learning repository, 2013. Dosegljivo: `http://archive.ics.uci.edu/ml`. [Dostopano 31.8.2015].

[27] Michael C. Mozer and Paul Smolensky. Advances in neural information processing systems 1. chapter Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment, pages 107–115. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[28] Behnam Neyshabur and Rina Panigrahy. Sparse matrix factorization. *arXiv preprint arXiv:1311.3315*, 2013.

[29] Yoh-Han Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] Foster J Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453, 1998.

[32] Quora. How does deep learning work and how is it different from normal neural networks and/or svm?, 2014. Dosegljivo: `http://www.quora.com` [Dostopano 22.8.2015].

[33] Russell Reed. Pruning algorithms-a survey. *Neural Networks, IEEE Transactions on*, 4(5):740–747, 1993.

[34] Eric Roberts. The intellectual excitement of computer science, 2000. Dosegljivo: `http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html`. [Dostopano 31.8.2015].

[35] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE, 2013.

[36] Tara N Sainath, Brian Kingsbury, Hagen Soltau, and Bhuvana Ramabhadran. Optimization techniques to improve training speed of deep neural networks for large speech tasks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(11):2267–2276, 2013.

[37] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[38] Joel A Tropp. Literature survey: Nonnegative matrix factorization. *University of Texas at Austin*, 2003.

[39] Guoqiang Peter Zhang. Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4):451–462, 2000.

[40] Yu Zhang, Ekapol Chuangsuwanich, and James Glass. Extracting deep neural network bottleneck features using low-rank matrix factorization. In *Proc. ICASSP*, pages 185–189, 2014.

[41] Marinka Žitnik. *Pristop matrične faktorizacije za gradnjo napovednih modelov iz heterogenih podatkovnih virov*. PhD thesis, Univerza v Ljubljani, 2012.

[42] Marinka Zitnik and Blaz Zupan. Data fusion by matrix factorization. *CoRR*, abs/1307.0803, 2013.

[43] Marinka Zitnik and Blaz Zupan. Data fusion by matrix factorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(1):41–53, 2015.